c-NTPP: Learning Cluster-Aware Neural Temporal Point Process

Fangyu Ding[1], **Junchi Yan**[1*], **Haiyang Wang**[2]

[1] Department of Computer Science and Engineering and MOE Key Lab of AI, Shanghai Jiao Tong University
[2] Ant Group, Hangzhou, China
{arthur_99, yanjunchi}@sjtu.edu.cn, chixi.why@mybank.cn

Abstract

Event sequences in continuous time space are ubiquitous across applications and have been intensively studied with both classic temporal point process (TPP) and its recent deep network variants. This work is motivated by an observation that many of event data exhibit inherent clustering patterns in terms of the sparse correlation among events, while such characteristics are seldom explicitly considered in existing neural TPP models whereby the history encoders are often embodied by RNNs or Transformers. In this work, we propose a c-NTPP (Cluster-Aware Neural Temporal Point Process) model, which leverages a sequential variational autoencoder framework to infer the latent cluster each event belongs to in the sequence. Specially, a novel event-clustered attention mechanism is devised to learn each cluster and then aggregate them together to obtain the final representation for each event. Extensive experiments show that c-NTPP achieves superior performance on both real-world and synthetic datasets, and it can also uncover the underlying clustering correlations.

1 Introduction

Temporal point process (TPP) (Rubin 1972; Hawkes 1971) has been a popular and principled tool for modeling and predicting event sequence data in continuous time space, e.g. user behavior sequences on social media platforms (Zhou, Zha, and Song 2013), asset management (Yan et al. 2013), personalized healthcare records (Choi et al. 2015), high frequency financial transactions (Bacry and Muzy 2014) and earthquake records (Ogata 1998). Inside the event sequence, various types of asynchronous events often exhibit complex dependencies on their history (Wu et al. 2018), making events modeling even challenging.

With the development of deep learning, the so-called neural temporal point process (NTPP) models (Xiao et al. 2017b,a) have been intensively devised for their high capacity for complex event sequence modeling. Popular neural backbones for sequential data are used in NTPP which often serve as encoders to capture the history, e.g. the recurrent neural networks (RNNs) (Du et al. 2016; Mei and Eisner 2017; Omi, Aihara et al. 2019; Shchur, Biloš, and Günnemann 2020), and attention mechanisms including the Transformer (Zhang et al. 2020; Zuo et al. 2020). On the other hand, a decoder is used to model the probability distribution of the occurrence time and type of the next event.

In this paper, we tend to make an (arguable) observation that many real-world event data often show inherent sparsity in terms of correlation or causality over the events, which can also be recognized from the event clustering perspective i.e. the correlations are within each cluster. Certainly, such observation has been also implicitly or explicitly used in literature. For example, the classic and widely adopted Hawkes process (Hawkes 1971) (also known as the self-exciting process) often uses a fast time decay kernel e.g. exponential kernel to model the dynamics of event sequence, which in fact exhibits a sparse influence on the current event from the history events. In practice, the sequence data is often not segmented based on the detection of large time intervals and subsequences, which also leads to sparse dependencies among events. In fact, sparsity is a ubiquitous nature in many areas, and it also has been widely adopted as an inductive bias not only to fit the data but also to mitigate the illness of model learning as a regularizer (Zhou, Zha, and Song 2013). Despite the recent efforts to such sparsity correlation pattern in the domain of time series (contrastively in the discrete time domain) with deep networks (Li et al. 2019; Zhou et al. 2021) especially for the Autoformer (Xu et al. 2021) that leverages a decomposition method based on the Auto-Correlation mechanism, little attention has been paid to event data, especially from the NTPP perspective. In fact, the currently dominant backbones used in NTPP like RNN (Du et al. 2016) or Transformer (Zuo et al. 2020) provide no tailored scheme to capture the sparsity (if there is).

To capture the possible clustering of event influence, we propose a novel representation learning framework for event sequence, namely Cluster-Aware Neural Temporal Point Process (c-NTPP). We decompose an event sequence into different little correlated clustered subsequences in the manner of sequential variational inference (Chung et al. 2015). A discrete sequential variational autoencoder (SVAE) is leveraged to infer the latent cluster each event belongs to through its encoder and decompose the event sequence into different clusters. As for the SVAE decoder part, with the latent clusters inferred, we formulate the probabilistic model for the decomposed temporal point process (DTPP) which is an

[*]Correspondence author is Junchi Yan who is also affiliated with Shanghai AI Laboratory, Shanghai, China.

analog to the reconstruction probability of a VAE. Moreover, unlike the recurrent units or global attention used to model the history events in existing NTPP models, we propose an event-clustered attention mechanism to get an augmented representation for each event based on each cluster.

The contributions of the paper are summarized as follows:

- To the best of our knowledge, we are the first to propose a deep event clustering strategy for TPP with SVAE.
- Based on the inferred latent clusters for events, we develop an event-clustered attention mechanism to improve the event sequence representations learning and also its interpretability, especially for the common event-clustered sequences e.g. the Hawkes process.
- Extensive experiments show the expressiveness of our proposed c-NTPP and the soundness of event clustering.

Further discussion. We give comments on its connection to related techniques and open problems beyond this paper.

1) Difference between sparse Transformer models (Zhou et al. 2021; Xu et al. 2021): In addition to the difference in the research area (time series v.s. temporal point process) and inner mechanisms (e.g. Autoformer's Auto-Correlation v.s. our sequential variational inference), it is worth noting that our c-NTPP is not a sparse Transformer model to reduce the computational complexity of Transformer architecture. We aim at a more expressive TPP representation learning which captures the sparsity of event sequences from the perspective of event clustering.

2) Handling dense and even correlation: In cases when the maximum number of clustering $K = 1$, the clustering structure does not exist, and then our model exactly degenerates to the THP (Zuo et al. 2020). In another word, THP is a special case of our proposed model.

3) The relation among event types: In line with many peer methods (Zuo et al. 2020), such relations are not specially treated in this paper, though some other works (Zhou, Zha, and Song 2013) pay more attention to this regard. We believe incorporating such type-level correlation sparsity into our model may be nontrivial which we leave for future work.

2 Preliminaries and Related Works

Before diving into details, we first introduce some preliminaries to facilitate the presentation of our approach.

2.1 Temporal Point Process

A TPP can be represented as a counting process $N(t)$ which counts the number of events that happened until time t. Given a sequence of events at times $\{t_i\}_{i=1}^{L}$, the TPP can be characterized by the conditional intensity function $\lambda(t|\mathcal{H}_t)$ reflecting the probability of the occurrence of an event in $[t, t+\mathrm{d}t)$ conditioned on the history $\mathcal{H}_t$:

$$\lambda(t|\mathcal{H}_t)\mathrm{d}t = \mathbb{P}(N(t+\mathrm{d}t) - N(t) = 1|\mathcal{H}_t), \tag{1}$$

where the notation of conditional intensity $\lambda(t|\mathcal{H}_t)$ is often simplified to $\lambda^*(t)$. The density can be derived as:

$$p^*(t) = \lambda^*(t) \exp\left(-\int_{t_{i-1}}^{t} \lambda^*(\tau)\mathrm{d}\tau\right). \tag{2}$$

A multi-dimensional TPP (MTPP) also considers the event type (a.k.a marker) of each event, each type m has a conditional intensity function $\lambda_m^*(t)$ accordingly, they sum up to the total intensity $\lambda^*(t)$.

Given the event sequence $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{L}$ of length L, where $\mathbf{x}_i = (t_i, m_i)$ represents the i-th event, $t_i \in [0, T]$ is the timestamp and $m_i \in \{1, ..., M\}$ is the marker, the MTPP can be learned via Maximum Likelihood Estimation (MLE). The log-likelihood of the sequence $\mathbf{X}$ over the time span $[0, T]$ is given by (Zhang et al. 2020):

$$\mathcal{L} = \sum_{i=1}^{L} \log \lambda_{m_i}^*(t_i) - \int_0^T \lambda^*(\tau)\mathrm{d}\tau. \tag{3}$$

2.2 Variational Autoencoder

A Variational Autoencoder (VAE) (Kingma and Welling 2013) is a generative model comprising of two parts: an inference sub-model and a generative sub-model. The inference part is an probabilistic encoder modeling the posterior distribution $q(\mathbf{z}|\mathbf{x})$ and maps the observed variables $\mathbf{x}$ to the latent variables $\mathbf{z}$ which approximate a prior $p(\mathbf{z})$. The generative part is a probabilistic decoder modeling the likelihood $p(\mathbf{x}|\mathbf{z})$ and reconstructs the visible variables $\mathbf{x}$ given the latent variables $\mathbf{z}$. A VAE model could either use a continuous prior (Kingma and Welling 2013) or a discrete prior (Van Den Oord, Vinyals et al. 2017). The VAE model can be efficiently trained by optimizing the evidence lower bound (ELBO) of $\log p(\mathbf{x})$:

$$\begin{aligned}\log p(\mathbf{x}) &\geq \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) \\ &= \mathcal{L}(q),\end{aligned} \tag{4}$$

The ELBO objective consists of an expectation item which is often approximated by sampling methods, and a KL divergence item which often has an analytic (Kingma and Welling 2013) or approximated (Nalisnick and Smyth 2016; Joo et al. 2020) solution.

2.3 Temporal Point Process Clustering

There are existing sequence clustering works and event clustering works discussing the clustering problem under the context of the temporal point process.

Sequence clustering aims at distinguishing the event sequences with different temporal patterns. For instance, the works (Wu et al. 2022; Xu and Zha 2017; Zhang et al. 2022) learn mixture models by reinforcement learning or mean-field variational inference to infer the latent clusters for event sequence clustering.

The topic of our cluster inference method belongs to event clustering. Unlike sequence clustering where different sequences are independent of each other, for event clustering, the goal is to cluster events in a sequence and there are complex dependencies among different events. We only identify (Yang and Zha 2013) which exactly shares the same TPP cluttering problem setting with ours, which leverages the mixture of the Hawkes process to model the influences among different events. Meanwhile, there exists other event clustering works, yet they all differ from our setting. The work (Li et al. 2014) is tailored to the online search setting, and additional text content information about the query is heavily explored (e.g. topic model) to achieve query clustering and user intention understanding. While our approach is purely based on time-stamp and marker. Moreover, our method is neural while (Li et al. 2014) still relies on the Hawkes process assumption which is questionable in many real-world cases as evidenced by previous works (Du et al. 2016; Omi, Aihara et al. 2019). The work (Kim et al. 2017) devises a topic model to do robust recommendation against missing observations (the 'silence' problem), different topics represent different temporal patterns of user behavior. However, in (Yang and Zha 2013) and ours, the temporal patterns (generating process) of different clusters are shared but events in different clusters are weakly correlated.

3 The Proposed Model

In this section, we introduce our Cluster-Aware Neural Temporal Point Process (c-NTPP) for the modeling of event sequences. More specifically, we infer the latent cluster each event belongs to and propose a probability model for the decomposed (clustered) event sequence. We provide a sequential variational autoencoder (SVAE) framework for end-to-end learning and an event-clustered attention mechanism to learn more expressive hidden representations for each event.

3.1 Cluster Inference for Events

We start by introducing some notations and the problem definition. Given an event sequence $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{L}$, our goal is to infer the latent variables $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{L}$ corresponding to each event, where each $\mathbf{z}_i$ is a discrete 1-of-K vector representing the latent cluster event $\mathbf{x}_i$ belongs to, i.e. $z_{ik} = 1$ *iff* event $\mathbf{x}_i$ belongs to the k-th cluster, and K is the predefined maximum cluster number. Each $\mathbf{z}_i$ is drawn from a Categorical posterior distribution parameterized by $\boldsymbol{\pi}_i^q$ learned from sequential information with a sequence-to-sequence model.

In line with the NTPP models, we first obtain the hidden representations for each event as:

$$\begin{aligned} \mathbf{e}_i &= \text{EventEmb}(\mathbf{x}_i), \\ \mathbf{h}_{1:L} &= \text{Seq2Seq}(\mathbf{e}_{1:L}), \end{aligned} \tag{5}$$

where we adopt the same design as the Transformer Hawkes Process (THP) for our event embedding module EventEmb and sequential encoder module Seq2Seq.

With the event embeddings and hidden representations obtained, under the SVAE framework, we formulate the posterior $q(\mathbf{z}_i|\mathbf{x}_{\le i})$ and prior $p(\mathbf{z}_i|\mathbf{x}_{<i})$ for each event as:

$$\boldsymbol{\pi}_i^q = \varphi^q(\mathbf{e}_i, \mathbf{h}_{i-1}), \tag{6}$$

$$\boldsymbol{\pi}_i^p = \varphi^p(\mathbf{h}_{i-1}), \tag{7}$$

$$q(\mathbf{z}_i|\mathbf{x}_{\le i}) = \text{Categorical}_K(\boldsymbol{\pi}_i^q), \tag{8}$$

$$p(\mathbf{z}_i|\mathbf{x}_{<i}) = \text{Categorical}_K(\boldsymbol{\pi}_i^p), \tag{9}$$

where φ_q and φ_p can be any flexible functions computing the posterior and prior distribution parameters $\boldsymbol{\pi}_i^q$ and $\boldsymbol{\pi}_i^p$ for each event. In c-NTPP, these two functions are given by:

$$\boldsymbol{\pi}_i^q = \text{softmax}(\mathbf{W}^q(\mathbf{e}_i + \mathbf{h}_{i-1}) + \mathbf{b}^q), \tag{10}$$

$$\boldsymbol{\pi}_i^p = [1/K, \dots, 1/K]^\top, \tag{11}$$

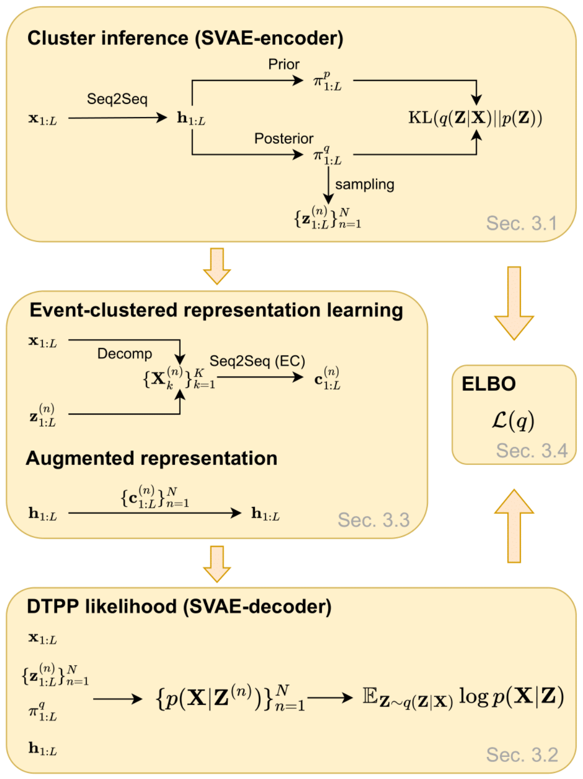

Cluster inference (SVAE-encoder)
Prior
$\pi^p_{1:L}$
Seq2Seq
$\mathbf{x}_{1:L}$
$\mathbf{h}_{1:L}$
$\mathrm{KL}(q(\mathbf{Z}|\mathbf{X})||p(\mathbf{Z}))$
Posterior
$\pi^q_{1:L}$
sampling
$\{\mathbf{z}^{(n)}_{1:L}\}_{n=1}^N$
Sec. 3.1
Event-clustered representation learning
$\mathbf{x}_{1:L}$
Decomp
Seq2Seq (EC)
$\{\mathbf{X}_k^{(n)}\}_{k=1}^K$
$\mathbf{c}^{(n)}_{1:L}$
$\mathbf{z}^{(n)}_{1:L}$
Augmented representation
$\{\mathbf{c}^{(n)}_{1:L}\}_{n=1}^N$
$\mathbf{h}_{1:L}$
$\mathbf{h}_{1:L}$
Sec. 3.3
ELBO
$\mathcal{L}(q)$
Sec. 3.4
DTPP likelihood (SVAE-decoder)
$\mathbf{x}_{1:L}$
$\{\mathbf{z}^{(n)}_{1:L}\}_{n=1}^N$
$\pi^q_{1:L}$
$\mathbf{h}_{1:L}$
$\{p(\mathbf{X}|\mathbf{Z}^{(n)})\}_{n=1}^N$
$\mathbb{E}_{\mathbf{Z}\sim q(\mathbf{Z}|\mathbf{X})} \log p(\mathbf{X}|\mathbf{Z})$
Sec. 3.2

Figure 1: Workflow of the proposed c-NTPP. Under the SVAE framework, a sequential encoder Seq2Seq learns the representations $\mathbf{h}_i$ (with self-attention mechanism consistent with THP) and $\mathbf{c}_i$ (with the event-clustered attention mechanism proposed in Sec. 3.3) for each event $\mathbf{x}_i$ in a sequence. The parameters $\boldsymbol{\pi}_{1:L}^q$ for the posterior and $\boldsymbol{\pi}_{1:L}^p$ for the prior are obtained as described in Sec. 3.1. With the sampled latent $\mathbf{Z}^{(n)}$'s from the posterior $q(\mathbf{Z}|\mathbf{X})$, the likelihood $p(\mathbf{X}|\mathbf{Z}^{(n)})$ used in the decoder of the SVAE is given by the likelihood of decomposed TPP (DTPP) in Sec. 3.2. The training objective of c-NTPP is the ELBO of SVAE, as discussed in Sec. 3.4.

where φ^q is a learnable function implemented with a linear transformation (parameterized by $\mathbf{W}^q$ and $\mathbf{b}^q$) followed by a softmax operation; and φ^p is a constant function for a uniform prior.

Therefore, the posterior $q(\mathbf{Z}|\mathbf{X})$ and prior $p(\mathbf{Z})$ for the entire sequence of length L are:

$$q(\mathbf{Z}|\mathbf{X}) = \prod_{i=1}^{L} \text{Categorical}_K(\boldsymbol{\pi}_i^q), \tag{12}$$

$$p(\mathbf{Z}) = \prod_{i=1}^{L} \text{Categorical}_K(\boldsymbol{\pi}_i^p). \tag{13}$$

3.2 Likelihood of Decomposed TPP

With each event's latent cluster identity inferred, an event sequence can be decomposed into K disjoint clusters. We denote the k-th cluster as $\mathbf{X}_k = \{\mathbf{x}_i\}_{i\in C_k}$, where C_k is event identity set of the k-th cluster. Given a cluster inference result $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^L$, based on the independent assumption that different clusters are weakly correlated, the conditional probability of such a TPP decomposition can be modeled as the product of the probabilities of K clusters:

$$p(\mathbf{X}|\mathbf{Z}) = \prod_{k=1}^{K} p(\mathbf{X}_k), \tag{14}$$

which is the conditional probability of the observed sequence $\mathbf{X}$ given the cluster inference result $\mathbf{Z}$ and the posterior distribution parameters $\{\boldsymbol{\pi}_i^q\}_{i=1}^L$. The substitution $\frac{\lambda_k^*(t_i)}{\lambda^*(t_i)} = \boldsymbol{\pi}_{ik}^q$ can be made to avoid modeling the conditional intensity function λ_k^* for K clusters respectively, and to reduce the complexity of the cluster inference task by K times. Eq.14 can be further derived as:

$$\begin{aligned}
&p(\mathbf{X}|\mathbf{Z}) = \prod_{k=1}^{K} p(\mathbf{X}_k) \\
&= \prod_{k=1}^{K} \left[\exp\left(-\int_0^T \lambda_k^*(\tau)\mathrm{d}\tau \right) \prod_{i \in C_k} \lambda_k^*(t_i) \right] \\
&= \left[\prod_{k=1}^{K} \exp\left(-\int_0^T \lambda_k^*(\tau)\mathrm{d}\tau \right) \right] \left[\prod_{k=1}^{K} \prod_{i \in C_k} \lambda_k^*(t_i) \right] \\
&= \exp\left(-\int_0^T \lambda^*(\tau)\mathrm{d}\tau \right) \prod_{i=1}^{L} \prod_{k=1}^{K} [\lambda_k^*(t_i)]^{z_{ik}} \\
&= \exp\left(-\int_0^T \lambda^*(\tau)\mathrm{d}\tau \right) \prod_{i=1}^{L} \prod_{k=1}^{K} [\lambda^*(t_i)\boldsymbol{\pi}_{ik}^q]^{z_{ik}} \\
&= \left[\exp\left(-\int_0^T \lambda^*(\tau)\mathrm{d}\tau \right) \prod_{i=1}^{L} \lambda^*(t_i) \right] \left[\prod_{i=1}^{L} \prod_{k=1}^{K} \boldsymbol{\pi}_{ik}^{q\ z_{ik}} \right],
\end{aligned} \tag{15}$$

the conditional probability $p(\mathbf{X}|\mathbf{Z})$ in Eq. 15 also serves as the likelihood function under our SVAE framework and it turns out to be a multiplication of a traditional TPP likelihood term of Eq.2 and a $\prod_{i=1}^L \prod_{k=1}^K \boldsymbol{\pi}_{ik}^{q\ z_{ik}}$ term. By optimizing the ELBO objective in Sec.3.4, the two terms form a self-modulating structure as jumps of the TPP likelihood indicate the occurrence of outlier events and the changes of the cluster property, which lead to a prediction guided latent inference of $\boldsymbol{\pi}_i^q$. We leave out the marker information for notation conciseness.

As a decoder of the hidden representations, the conditional intensity function λ^* can be any highly flexible function. We use a neural network (NN) decoder in c-NTPP:

$$\lambda(t|\mathbf{h}_i) = \mathrm{Softplus}(\mathrm{NN}(\Phi(t - t_i) + \mathbf{h}_i)), \tag{16}$$

where the time encoding function Φ adopts the same design in (Zuo et al. 2020), encoding the interval $t - t_i$, and the softplus function ensures the intensity value positive.

We compute the non-event log-likelihood $\int_{t_{i-1}}^{t_i} \lambda^*(\tau)\mathrm{d}\tau$ by Monte Carlo integration (Robert and Casella 1999).

3.3 Event-Clustered Attention Mechanism

To deal with the potential sparsity in event sequence data and learn more expressive event representations, we propose the event-clustered attention mechanism based on the decomposed TPP obtained above. Unlike the baselines regarding the history as a whole, we take advantage of the decomposed clusters according to a cluster inference result $\mathbf{Z}$ for the representation learning for each event. The architecture of the event-clustered attention mechanism is shown in Fig. 2.

The input of an event-clustered attention layer is a target event information $\mathbf{h}_i^{(l-1)}$ and the history information of each cluster $\{\mathcal{H}_{ik}\}_{k=1}^K$, where $\mathcal{H}_{ik} = \{\mathbf{h}_j^{(l-1)}\}_{j \in C_k \wedge j \leq i}$, when $l = 1$, i.e. for the first layer, the inputs are just the event embeddings. For each cluster k, we first obtain the 'query', 'key' and 'value' according to the self-attention mechanism:

$$\mathbf{q}_i = \mathbf{h}_i^{(l-1)}\mathbf{W}_Q,\ \mathbf{K}_{ik} = \mathbf{H}_{ik}\mathbf{W}_K,\ \mathbf{V}_{ik} = \mathbf{H}_{ik}\mathbf{W}_V, \tag{17}$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$ and $\mathbf{W}_V$ are the weight matrices, $\mathbf{H}_{ik}$ is the matrix version of $\mathcal{H}_{ik}$ (i.e. the history event representation set of cluster k).

In line with the self-attention mechanism, we then obtain the influence from the history information of the k-th cluster for the i-th event as:

$$\mathbf{c}_{ik} = \mathrm{Attn}(\mathbf{q}_i, \mathbf{K}_{ik}, \mathbf{V}_{ik}). \tag{18}$$

The final event-clustered representation $\mathbf{c}_i$ is obtained by aggregating these K representations corresponding to each cluster with a certain aggregator Agg_K (e.g. mean aggregator, attention aggregator, etc.):

$$\mathbf{c}_i = \mathrm{Agg}_K(\{\mathbf{c}_{ik}\}_{k=1}^K). \tag{19}$$

where we adopt the design of the AttSets (Yang et al. 2020) for the attention aggregator in our work.

Under the SVAE framework, we could have N cluster inference results sampled from the posterior, i.e. $\forall n \in \{1, \ldots, N\}, \mathbf{Z}^{(n)} \sim q(\mathbf{Z}|\mathbf{X})$, we can learn N event-clustered representations $\{\mathbf{c}_i^{(n)}\}_{n=1}^N$ for each event, we aggregate them again with aggregator Agg_N and obtain the final event-clustered representation:

$$\mathbf{c}_i = \mathrm{Agg}_N(\{\mathbf{c}_i^{(n)}\}_{n=1}^N), \tag{20}$$

in practice, we can avoid the design of Agg_N by setting the sample number $N = 1$.

Finally, by combining the representations $\mathbf{h}_i$ and $\mathbf{c}_i$ together (there are other possible ways such as averaging, attentive aggregation, and concatenation. We use adding by simplicity), the representation $\mathbf{h}_i$ obtained from the sequential model Seq2Seq (Eq. 5) for each event $\mathbf{x}_i$ is augmented by the event-clustered representation $\mathbf{c}_i$:

$$\mathbf{h}_i := \mathbf{h}_i + \mathbf{c}_i, \tag{21}$$

where $\mathbf{h}_i$ and $\mathbf{c}_i$ can be obtained from the same Seq2Seq encoder. We do not use additional parameters for $\mathbf{c}_i$ except for the attention aggregators in the event-clustered attention mechanism. When $K = 1$, c-NTPP degenerates to THP with global attention and all the history events belong to a single cluster.

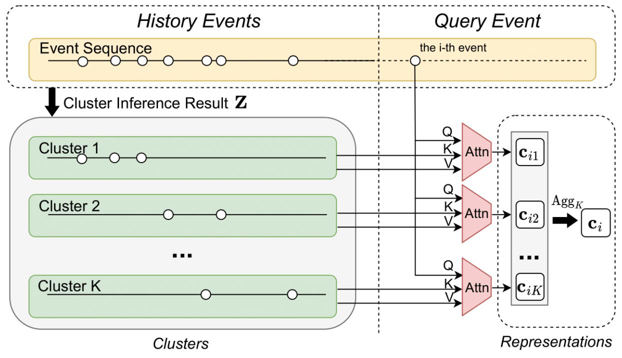


Figure 2: The event-clustered attention mechanism for TPP modeling where the event sequence is decomposed into K clusters according to a cluster inference result $\mathbf{Z}$. For each event, K number of representations $\mathbf{c}_{ik}$ are learned for each cluster and then aggregated to the final representation $\mathbf{c}_i$.

3.4 End-to-End Gradient-Based Training

As shown in Fig. 1, our proposed c-NTPP is trained under the SVAE framework and the training objective is the ELBO (as described in Eq. 4). Based on the definitions of our posterior Eq. 12 and prior Eq. 13, the KL divergence item in the ELBO can be derived as:

$$\mathrm{KL}[q(\mathbf{Z}|\mathbf{X})\|p(\mathbf{Z})] = \sum_{i=1}^{L}\sum_{k=1}^{K} \boldsymbol{\pi}_{ik}^{q} \log \frac{\boldsymbol{\pi}_{ik}^{q}}{\boldsymbol{\pi}_{ik}^{p}}. \quad (22)$$

Using Monte Carlo expectation and taking N samples of the latent variables $\mathbf{Z}^{(n)} \sim q(\mathbf{Z}|\mathbf{X})$, the ELBO is approximated as below and can be end-to-end learned under the SVAE framework via gradient descent:

$$\mathcal{L}(q) = \frac{1}{N}\sum_{n=1}^{N} \log p(\mathbf{X}|\mathbf{Z}^{(n)}) - \mathrm{KL}[q(\mathbf{Z}|\mathbf{X})\|p(\mathbf{Z})]. \quad (23)$$

The procedure of a forward pass of c-NTPP on an event sequence is shown in Alg. 1. The input is the event sequence $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{L}$. After the phases of cluster inference, sampling of latent $\mathbf{Z}^{(n)}$, representation augmentation with the event-clustered attention mechanism, and computation of the training objective ELBO, the output is the representations of the events $\{\mathbf{h}_i\}_{i=1}^{L}$, the probabilities each event belongs to each cluster $\{\boldsymbol{\pi}_i^{q}\}_{i=1}^{L}$, and the ELBO training objective $\mathcal{L}$. The modules in c-NTPP can be end-to-end learned under the SVAE framework via gradient descent.

Algorithm 1: The forward pass of c-NTPP.

Input: The event sequence $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{L}$.
Parameter: The maximum cluster number K. The sampling times number N.
Output: The representations $\{\mathbf{h}_i\}_{i=1}^{L}$. The latent cluster distributions $\{\boldsymbol{\pi}_i^{q}\}_{i=1}^{L}$. The ELBO training objective $\mathcal{L}$.

```
 1: $\mathbf{e}_{1:L}$ = EventEmb($\mathbf{x}_{1:L}$);
 2: $\mathbf{h}_{1:L}$ = Seq2Seq($\mathbf{e}_{1:L}$);
    // Cluster inference
 3: $\boldsymbol{\pi}^{q}_{1:L} = \varphi^{q}(\mathbf{e}_i, \mathbf{h}_{i-1})$;
 4: $\boldsymbol{\pi}^{p}_{1:L} = \varphi^{p}(\mathbf{h}_{i-1})$;
    // Latent cluster sampling
 5: for n = 1,...,N do
 6:     $\mathbf{Z}^{(n)} = \mathbf{z}^{(n)}_{1:L} \sim \mathrm{Categorical}_K(\boldsymbol{\pi}^{q}_{1:L})$;
 7: end for
    // Event-clustered augmented representations
 8: for n = 1,...,N do
 9:     $\mathbf{c}^{(n)}_{1:L} = \mathrm{Seq2Seq_{EC}}(\mathbf{X}, \mathbf{Z}^{(n)})$;
10: end for
11: $\mathbf{h}_{1:L} = \mathbf{h}_{1:L} + \mathrm{Agg}_N(\{\mathbf{c}^{(n)}_{1:L}\}_{n=1}^{N})$;
    // DTPP likelihood
12: for n = 1,...,N do
13:     $p(\mathbf{X}|\mathbf{Z}^{(n)}) = \prod_{i=1}^{L} \boldsymbol{\pi}^{q}_{i,\mathbf{z}_i^{(n)}} \lambda^*(t_i) \exp(-\int_0^T \lambda^*(\tau)\mathrm{d}\tau)$
14: end for
    // The ELBO training objective
15: $\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N} \log p(\mathbf{X}|\mathbf{Z}^{(n)}) - \mathrm{KL}[q(\mathbf{Z}|\mathbf{X})\|p(\mathbf{Z})]$
16: return $\{\mathbf{h}_i\}_{i=1}^{L}$, $\{\boldsymbol{\pi}_i^{q}\}_{i=1}^{L}$, $\mathcal{L}$
```

4 Experiments

Experiments are conducted on both synthetic and real-world datasets. Evaluation metrics include fitting likelihood and predictive performance for representation learning and its ability in terms of latent cluster inference. Ablation studies are also performed to compare our proposed event-clustered attention mechanism and its alternative aggregation schemes. We further analyze the latent cluster inference result on our synthetic datasets which have latent clusters.

4.1 Datasets and Protocols

1) SYN. A synthetic dataset simulated by the open-source library Tick (Bacry et al. 2017). We simulate a 2-D Hawkes process whose conditional intensity function is specified by $\lambda_m(t) = \mu_m + \sum_{t_i<t} \alpha_{mm_i}\beta \exp(-\beta(t-t_i))$ where we set the base intensities $\mu_0 = \mu_1 = \frac{1}{300}$, the kernel intensities $\alpha_{00} = \alpha_{11} = 0.5$, $\alpha_{01} = \alpha_{10} = 0.1$ and the decays $\beta = 0.5$. The time window is set as $T = 100$ for each sequence. We construct SYN dataset by randomly selecting event sequence pairs and mixing them together to get independent subsequences (clusters) in each record.
2) FIN. The financial dataset (Du et al. 2016) contains high-frequency transaction records collected from NYSE for a stock in one day, with 0.7 million transactions. Each transaction contains the timestamp (in milliseconds) and possible action (buy/sell). The input data is a single long sequence and we cut it into pieces for training and testing. The action type is a marker and we predict when an action occurs.
3) MMC. MIMIC II (Saeed et al. 2002) collects de-identified clinical visit records of ICU patients for 7 years. Each patient has a sequence of hospital visit events recording the timestamps and their disease types. It contains 75 disease types as event markers.
4) RET. The Retweets dataset (Zhao et al. 2015) contains sequences of tweets. Each sequence contains an origin tweet (i.e., a user initiates a tweet), and a subsequent stream of

	Encoder	#layers	Decoder
RMTPP	RNN	2	EXP
NHP	RNN	2	HP
FullyNN	RNN	2	NN
SAHP	TFM	4	HP
THP	TFM	4	NN
c-NTPP-n	VTFM	4	NN
c-NTPP	VTFM-EC	4	NN

Table 1: Model architecture comparison.

retweets. The timestamps and the user types are recorded, and the users are grouped into three categories based on the number of their followers: "small", "medium", and "large".
5) SO. The StackOverflow dataset (Paranjape, Benson, and Leskovec 2017) contains two years of user awards on the question-answering website. Each user received a sequence of badges (of 22 different types) and the reward history of each user is treated as the event sequence.

4.2 Compared Models

The compared models are the state-of-the-art baselines and several variants of our proposed c-NTPP. We summarize the differences between these TPP models in terms of the encoder and the decoder architectures in Tab. 1, the column of 'Encoder' refers to the Seq2Seq module used to obtain the representations of the events, the column of '# layers' refers to the layer number of the encoders, and the column of 'Decoder' refers to the form of the conditional intensity function which decodes the representations of the events. The 'HP' decoders refer to the intensity forms designed similar to that of the Hawkes process.
1) RMTPP (Du et al. 2016) uses RNN to encode the history information from past events, and uses an exponential function as the decoder to model the conditional intensity.
2) NHP (Mei and Eisner 2017) uses a continuous-time LSTM to model self-modulating Hawkes processes and can get a continuous-time representation for imitating.
3) FullyNN (Omi, Aihara et al. 2019) models the cumulative intensity function (i.e. $\int_{t_{i-1}}^{t} \lambda^*(\tau)\mathrm{d}\tau$) with a neural network monotonic increasing for t and takes advantage of the autograd mechanism of deep learning frameworks to obtain the conditional intensity function.
4) SAHP (Zhang et al. 2020) uses the self-attention mechanism to develop a sequential encoder for TPP and models the intensity similar to the form of the Hawkes process.
5) THP (Zuo et al. 2020) leverages the Transformer encoder for TPP representation learning. In the original work, the intensity is parameterized by only one linear layer, which results in an intensity monotonic with time. Here we replace it with a 2-layer NN decoder, which is a more general decoder and also consistent with our c-NTPP model.
6) c-NTPP-n is c-NTPP with no event-clustered attention mechanism, i.e. without the augmentation in Eq. 21. A classic Transformer encoder (global attention) is leveraged to learn the posterior, prior, and intensity under the SVAE

Datasets	SYN	FIN	MMC	RET	SO
RMTPP	3.114	-	4.535	8.995	3.756
NHP	3.025	1.387	2.659	8.256	3.019
FullyNN	2.714	1.346	2.902	8.015	2.939
SAHP	2.593	1.297	2.652	7.270	2.831
THP	2.597	1.326	2.596	7.419	2.851
c-NTPP-n	2.610	1.195	2.665	7.379	2.571
c-NTPP-m	2.558	**1.173**	**2.138**	7.400	**2.551**
c-NTPP-a	**2.557**	1.180	2.178	**7.234**	2.692

Table 2: Negative log-likelihood per event on synthetic and real-world datasets. '-' denotes the model fails to converge.

framework, we abbreviate this encoder as VTFM (variational transformer) in Tab. 1.
7) c-NTPP is our proposed model, whose encoder is a VTFM equipped with the event-clustered attention mechanism, we abbreviate it as VTFM-EC in Tab. 1. We further compare the performance of the mean Agg_K (**c-NTPP-m**) versus the attention Agg_K (**c-NTPP-a**).

4.3 Training Details

In general and without fine-tuning or cherry picking, we uniformly set the hidden size to 64 in RNN encoders, the model dimension to 64, the feedforward network dimension to 128, and the number of attention heads to 4 in Transformer encoders. The number of layers is set to 2 for RNN encoders and 4 for Transformer encoders. Batch size is set to 16 and SVAE sample number N is set to 1. The number of maximum clusters is pre-defined as 4. We set the dropout rate as 0.1. The random seed is fixed to 42 for reproducible results.

We use Adam optimizer with a learning rate of 0.0001 for training. All the experiments run on a single RTX-2080Ti (11GB) GPU, we cut the long sequences to a maximum sequence length of 128 to avoid the GPU memory overflow. The pre-processing and the train-val-test split details of the datasets are consistent with previous works (Mei and Eisner 2017; Zuo et al. 2020), the validation datasets are used to find the optimal number of epochs to train on the training datasets, which alleviates the overfitting problem.

4.4 Log-Likelihood Evaluation

In line with the recent TPP learning works (Mei and Eisner 2017; Zhang et al. 2020; Zuo et al. 2020), we first use the per-event negative log-likelihood (NLL) as the metric for event sequence fitting (the lower the better).

We use the total conditional intensity learned from our proposed c-NTPP to obtain the NLL although we use the ELBO objective for training under the SVAE framework.

Tab. 2 reports the per-event NLL of these models on each test set, we can see that c-NTPP outperforms other baselines and c-NTPP-n by large margins, which demonstrates the expressiveness of the event-clustered attention mechanism on the TPP sequence data compared to the global attention mechanism used in SAHP, THP and c-NTPP-n as well as the RNN encoders used in RMTPP, NHP and FullyNN.

Algorithm 2: Median Absolute Deviation (MAD) to detect outliers and reduce the impact of huge time intervals.

Input: the time interval sequence $seq = [\Delta t_1, \ldots, \Delta t_L]$.
Parameter: the scaling factor n.
Output: indices of the outliers in sequence.

```
1: $median$ = median($seq$)
2: $mad$ = mean(abs($seq$ - $median$))
3: $indices$ = find(abs($seq$ - $median$) $> n \times mad$)
4: **return** $indices$
```

4.5 Predictive Ability Analysis

For prediction, based on the conditional intensity function $\lambda(t|\mathbf{h}_i)$ computed from the representation $\mathbf{h}_i$ of the i-th event, the next event time prediction can be given by:

$$\hat{t}_{i+1} = \int_{t_i}^{\infty} t \cdot p(t|\mathbf{h}_i)\mathrm{d}t, \tag{24}$$

where the conditional density function $p(t|\mathbf{h}_i)$ is calculated from the conditional intensity function $\lambda(t|\mathbf{h}_i)$ by Eq. 2. The integration is computed by Monte Carlo integration. To obtain a more reasonable prediction error evaluation, in a similar spirit to the filtering techniques in literature, we in this paper pay attention to those large time intervals between events e.g. in the Retweets dataset which are very difficult to predict as they are often rare events for existing TPP models. Hence the prediction metric is computing by excluding these long interval events which is fulfilled by our devised median absolute deviation (MAD) algorithm in Alg. 2 with the scaling factor $n = 10$.

Note that as we cannot sample to infinity, the integration upper limit in Eq. 24 by Monte Carlo integration is set to t_i added by twice the average of filtered time interval values $2\overline{\Delta t}$ on the training set in substitution for infinity. The median absolute deviation is applied for all datasets.

Note that although the prediction of these events are not counted in the performance evaluation, these events are still included in the training process and also as the input of the prediction model. This is a mask scheme for evaluation and inherently has nothing to do with data pre-processing for both training and prediction.

The next event marker prediction is:

$$\hat{m}_{i+1} = \underset{m \in \{1,\ldots,M\}}{\arg\max}\ \lambda_m(\hat{t}_{i+1}|\mathbf{h}_i), \tag{25}$$

where $\hat{t}_{i+1}$ is the predicted timestamp for next event and the subscript m refers to the discrete marker value in $\{1, \ldots, M\}$ (Note the difference from the cluster subscript k in Sec. 3.2). The evaluation metric is Root Mean Square Error (RMSE) for event time prediction and accuracy for event marker prediction. The results are summarized in Tab. 3 and Tab. 4, showing that our three c-NTPP models outperform the baselines on the event prediction tasks.

4.6 Ablation Studies

Results in Tab. 2, 3 and 4 show that c-NTPP-m and c-NTPP-a both stably outperform the baselines (including c-NTPP-n) except for the RMSE on Retweets dataset, which demonstrates the expressiveness of our event-clustered attention mechanism compared to the global attention mechanism used in c-NTPP-n and THP. Besides, c-NTPP-n and THP again show competitive performance, indicating the architecture of Transformer encoders used in c-NTPP-n and THP are of similar expressiveness.

Datasets	SYN	FIN	MMC	RET	SO
RMTPP	1.532	-	5.658	96.52	9.125
NHP	1.528	11.38	5.539	94.98	8.889
FullyNN	1.519	**11.35**	5.512	95.62	8.661
SAHP	1.591	11.37	5.579	93.35	8.757
THP	1.517	**11.35**	5.510	93.03	8.624
c-NTPP-n	1.521	11.36	5.519	**91.86**	8.650
c-NTPP-m	1.516	**11.35**	5.506	92.42	8.661
c-NTPP-a	**1.513**	**11.35**	**5.503**	92.71	**8.541**

Table 3: RMSE of next event time prediction.

Datasets	SYN	FIN	MMC	RET	SO
RMTPP	51.13	51.54	40.11	49.08	41.86
NHP	50.87	59.16	45.15	46.43	41.85
FullyNN	53.50	61.73	49.33	52.58	42.47
SAHP	**75.26**	62.55	57.46	**59.15**	43.15
THP	**75.26**	62.55	59.88	58.89	42.95
c-NTPP-n	**75.26**	**62.63**	59.88	58.59	43.23
c-NTPP-m	**75.26**	62.59	**73.25**	58.56	**43.86**
c-NTPP-a	**75.26**	62.61	69.18	59.13	43.51

Table 4: Next event marker prediction accuracy comparison.

4.7 Latent Cluster Inference Analysis

We construct two synthetic datasets to evaluate the clustering ability of our method. One is directly concatenate two independent sequences without overlapping, while the other is with overlapping. The sequences are generated the Hawkes process used in the SYN dataset in Sec. 4.1.

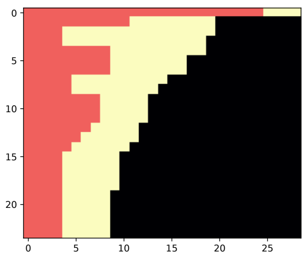
1) SEP. We simulate 5,000 event sequences by the Hawkes process and randomly select 2,500 pairs of event sequences. Each event sequence in the separated (SEP) dataset is constructed by concatenating the latter sequence in each pair after the former one, and the timestamp of each event in the latter sequence is added by the '10x' time span $10 \times T = 1000$ for the Hawkes process simulation.

2) OVL. The overlapped (OVL) dataset does not modify the timestamps in the two event sequences. It just mixes them together and then sorts the resulting sequence by the timestamp. OVL is the same as SYN used in the event modeling and prediction experiments.

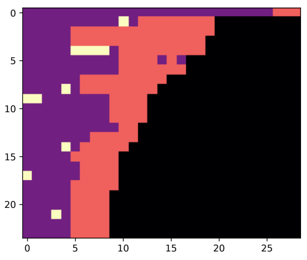
Fig. 3 visualizes the latent cluster inference result, where the inferred cluster for event $\mathbf{x}_i$ is obtained by:

$$\hat{k}_i = \underset{k \in \{1,\ldots,K\}}{\arg\max}\ \boldsymbol{\pi}^q_{ik}, \tag{26}$$

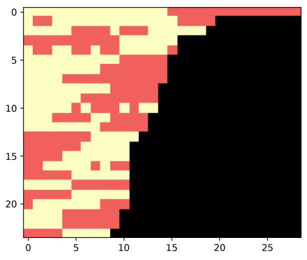
where the $\boldsymbol{\pi}^q_i$ is the posterior distribution parameter as described in Eq. 6, which represents the cluster assignment



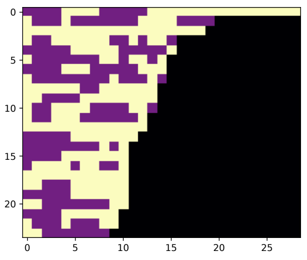
(a) SEP Ground Truth



(b) SEP Inference



(c) OVL Ground Truth



(d) OVL Inference

Figure 3: Latent cluster inference with c-NTPP(-a) on SEP and OVL. Each row denotes an event sequence in the datasets. The colors indicate which cluster the events belong to and the black region is the padding positions for each event sequence. The left plots are the ground truths of the SEP and OVL datasets respectively; the right plots are the inference results for these two datasets produced by c-NTPP.

Datasets	SEP	OVL
MHP (Yang and Zha 2013)	0.8044	0.1392
c-NTPP	**0.9837**	**0.4231**

Table 5: Latent cluster inference on SEP and OVL datasets in terms of Normalized Mutual Information (NMI).

probabilities for event $\mathbf{x}_i$.

We observe that different clusters can be clearly separated on the SEP dataset. However, the demonstration on the OVL dataset is relatively chaotic. Therefore, we use the Normalized Mutual Information (NMI) metric to evaluate the quality of the cluster inference by comparing the ground truth clusters and the inferred clusters. As shown in Tab. 5, the NMI values for c-NTPP exceed that with the baseline MHP (Yang and Zha 2013) on both datasets, indicating the improvement of the cluster inference quality.

The inference is performed by a single SVAE encoder forward pass for both c-NTPP and MHP (Yang and Zha 2013). Note that the work MHP (Yang and Zha 2013) is not open-sourced, and due to its complexity, we here try to implement a variant in the between, which aims to mimic MHP's principles. Instead of using the mean-field variational inference in MHP, we train MHP's model parameters by directly optimizing the ELBO objective under the SVAE framework, and event features are obtained by event embedding as done in the THP method (Zuo et al. 2020).

Dataset	# event	$\overline{\text{LEN}}$	MAD	DF
FIN	414800	2074	0.046	0.454
MMC	2419	4	0.155	0.139
SO	480413	72	0.580	0.260
RET	2173533	109	38.250	0.276
SEP	59953	12	1.082	0.075
OVL	59953	12	1.012	0.162

Table 6: Datasets statistics. $\overline{\text{LEN}}$: average sequence length; MAD: median absolute derivation; DF: degree of fluctuation of cluster inference results provided by c-NTPP.

4.8 Maximum Cluster Number Analysis

The cluster inference result for a sequence is related to the maximum cluster number K. When we set $K = 1$, the model architecture of c-NTPP exactly degenerates to that of THP, all the events in a sequence belong to a single cluster and the event-clustered attention has the same effect as the global attention in THP. More latent clusters might be inferred with a larger K, and the trade-off is to bring more computational complexity and variance. Fig. 3 also shows that an appropriate cluster number for each sequence could be learned with c-NTPP adaptively (i.e. there might be redundant clusters) by optimizing the ELBO objective.

4.9 Dataset Analysis

We summarize the datasets statistics in Tab. 6, for each dataset, we record the number of events (# event), average sequence length ($\overline{\text{LEN}}$), median absolute derivation (MAD) value of time intervals and the degree of fluctuation (DF) of cluster inference results provided by c-NTPP. A higher DF corresponds to a more chaotic clustering pattern, for exploratory analysis on latent clusters, we give an example of the above first FIN dataset consisting of high-frequency transactions, note it has the highest DF score among all 6 datasets, indicating it is hard to predict and has less obvious clustering pattern. We think the reason why our degenerated version outperforms SOTA transformer-based models a little, is two-fold: 1) there has been rich TPP literature, and the SOTA performance tends to saturate; 2) there is perhaps little clustering pattern in this testing data to boost prediction. As there is no cluster label for real-world data, we introduce the degree of fluctuation (DF) for cluster inference to measure the clustering pattern inside our 6 datasets: a higher DF corresponds to a more chaotic clustering pattern.

5 Conclusion

In this paper, we have proposed c-NTPP, a TPP representation learning framework which is able to capture the sparsity of event sequence data from the perspective of event clustering. Based on the cluster inference result under the SVAE framework, the event-clustered attention mechanism is leveraged to obtain a more expressive representation for each event. Experiments show the superiority of our c-NTPP against state-of-the-arts as well as the ability to uncover the underlying clustering correlations in event sequences.

Acknowledgements

This work was partly supported by National Natural Science Foundation of China (U19B2035, 61972250, 62222607), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and fund from Ant Group.

References

Bacry, E.; Bompaire, M.; Gaïffas, S.; and Poulsen, S. 2017. tick: a Python library for statistical learning, with a particular emphasis on time-dependent modeling. *arXiv preprint arXiv:1707.03003*.

Bacry, E.; and Muzy, J.-F. 2014. Hawkes model for price and trades high-frequency dynamics. *Quantitative Finance*, 14(7): 1147–1166.

Choi, E.; Du, N.; Chen, R.; Song, L.; and Sun, J. 2015. Constructing disease network and temporal progression model via context-sensitive Hawkes process. In *ICDM*.

Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A. C.; and Bengio, Y. 2015. A recurrent latent variable model for sequential data. *NeurIPS*.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *SIGKDD*.

Hawkes, A. 1971. Point spectra of some mutually exciting point processes. *Journal of the Royal Statistical Society. Series B (Methodological)*, 438–443.

Joo, W.; Lee, W.; Park, S.; and Moon, I.-C. 2020. Dirichlet variational autoencoder. *Pattern Recognition*, 107: 107514.

Kim, H.; Iwata, T.; Fujiwara, Y.; and Ueda, N. 2017. Read the Silence: Well-Timed Recommendation via Admixture Marked Point Processes. In *AAAI*.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Li, L.; Deng, H.; Dong, A.; Chang, Y.; and Zha, H. 2014. Identifying and labeling search tasks via query-based hawkes processes. *SIGKDD*.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *NeurIPS*.

Mei, H.; and Eisner, J. M. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. In *NeurIPS*.

Nalisnick, E.; and Smyth, P. 2016. Stick-breaking variational autoencoders. *arXiv preprint arXiv:1605.06197*.

Ogata, Y. 1998. Space-time point-process models for earthquake occurrences. *Annals of the Institute of Statistical Mathematics*.

Omi, T.; Aihara, K.; et al. 2019. Fully neural network based model for general temporal point processes. In *NeurIPS*.

Paranjape, A.; Benson, A. R.; and Leskovec, J. 2017. Motifs in Temporal Networks. In *WSDM*.

Robert, C. P.; and Casella, G. 1999. *Monte Carlo statistical methods*. Springer.

Rubin, I. 1972. Regular point processes and their detection. *IEEE Transactions on Information Theory*, 18(5): 547–557.

Saeed, M.; Lieu, C.; Raber, G.; and Mark, R. G. 2002. MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring. In *Computers in cardiology*.

Shchur, O.; Biloš, M.; and Günnemann, S. 2020. Intensity-Free Learning of Temporal Point Processes. In *ICLR*.

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. In *NeurIPS*, volume 30.

Wu, W.; Yan, J.; Yang, X.; and Zha, H. 2018. Decoupled Learning for Factorial Marked Temporal Point Processes. In *SIGKDD*.

Wu, W.; Yan, J.; Yang, X.; and Zha, H. 2022. Discovering temporal patterns for event sequence clustering via policy mixture model. *IEEE TKDE*, 34(2): 573–586.

Xiao, S.; Farajtabar, M.; Ye, X.; Yan, J.; Song, L.; and Zha, H. 2017a. Wasserstein Learning of Deep Generative Point Process Models. In *NIPS*.

Xiao, S.; Yan, J.; Yang, X.; Zha, H.; and Chu, S. 2017b. Modeling The Intensity Function Of Point Process Via Recurrent Neural Networks. In *AAAI*.

Xu, H.; and Zha, H. 2017. A Dirichlet Mixture Model of Hawkes Processes for Event Sequence Clustering. In *NeurIPS*.

Xu, J.; Wang, J.; Long, M.; et al. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*.

Yan, J.; Wang, Y.; Zhou, K.; Huang, J.; Tian, C.; Zha, H.; and Dong, W. 2013. Towards Effective Prioritizing Water Pipe Replacement and Rehabilitation. In *IJCAI*.

Yang, B.; Wang, S.; Markham, A.; and Trigoni, N. 2020. Robust attentional aggregation of deep feature sets for multi-view 3D reconstruction. *IJCV*, 128(1): 53–73.

Yang, S.-H.; and Zha, H. 2013. Mixture of mutually exciting processes for viral diffusion. In *ICML*.

Zhang, Q.; Lipani, A.; Kirnap, O.; and Yilmaz, E. 2020. Self-attentive Hawkes process. In *ICML*.

Zhang, Y.; Yan, J.; Zhang, X.; Zhou, J.; and Yang, X. 2022. Learning mixture of neural temporal point processes for event sequence clustering. In *IJCAI*.

Zhao, Q.; Erdogdu, M. A.; He, H. Y.; Rajaraman, A.; and Leskovec, J. 2015. Seismic: A self-exciting point process model for predicting tweet popularity. In *SIGKDD*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*.

Zhou, K.; Zha, H.; and Song, L. 2013. Learning Social Infectivity in Sparse Low-rank Networks Using Multi-dimensional Hawkes Processes. In *AISTATS*.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer Hawkes Process. In *ICML*.